SQA-Profiles: Rule-based Activity Profiles for Continuous Integration Environments

Brandtner, Martin ; Müller, Sebastian C ; Leitner, Philipp ; Gall, Harald C

DOI: https://doi.org/10.1109/SANER.2015.7081840

SQA-Profiles: Rule-Based Activity Profiles for Continuous Integration Environments

Martin Brandtner, Sebastian C. Müller, Philipp Leitner, and Harald C. Gall
University of Zurich, Department of Informatics, Switzerland
{brandtner, smueller, leitner, gall}@ifi.uzh.ch

***Abstract*—Continuous Integration (CI) environments cope with the repeated integration of source code changes and provide rapid feedback about the status of a software project. However, as the integration cycles become shorter, the amount of data increases, and the effort to find information in CI environments becomes substantial. In modern CI environments, the selection of measurements (e.g., build status, quality metrics) listed in a dashboard does only change with the intervention of a stakeholder (e.g., a project manager). In this paper, we want to address the shortcoming of static views with so-called Software Quality Assessment (SQA) profiles. *SQA-Profiles* are defined as rule-sets and enable a dynamic composition of CI dashboards based on stakeholder activities in tools of a CI environment (e.g., version control system). We present a set of SQA-Profiles for project management committee (PMC) members: Bandleader, Integrator, Gatekeeper, and Onlooker. For this, we mined the commit and issue management activities of PMC members from 20 Apache projects. We implemented a framework to evaluate the performance of our rule-based SQA-Profiles in comparison to a machine learning approach. The results showed that project-independent SQA-Profiles can be used to automatically extract the profiles of PMC members with a precision of 0.92 and a recall of 0.78.**

I. Introduction

Software development has become a data-driven discipline [1] and the tools used for Continuous Integration (CI) are important data sources in the development life cycle. The way of accessing data from CI environments differs between the stakeholders of a software project. For example, developers primarily perceive the CI-process in case of build exceptions (e.g., build or test failure), whereas software managers actively consolidate CI environments to gather data for planning and decision making purposes. The term *CI environment* in the context of our work refers to all platforms that are involved to perform and manage the automatic integration of source code changes. Such an environment typically consists of a version control system (VCS) and a issue tracking platform as well as other tools.

In earlier work [2], [3], we introduced a data integration approach for CI-data called SQA-Mashup. Our study showed that the proposed role-based tailoring fosters the interpretation of CI-data in a fast and accurate way. However, the composition and tailoring of the different views in state-of-the-art CI-tools as well as in SQA-Mashup is rather time-consuming and needs to be done by a professional. We propose activity data mining to overcome this shortcoming for enabling a fully-automatic composition of views, and a tailoring of CI-data according to the activities of a stakeholder. The use of the mined activity data is not restricted to visualization of CI-data. Additionally, it can also be used for project management purposes, such as workload reporting.

In this work, we propose a rule-based approach to automatically profile stakeholders based on their activities in the version control system (VCS) and the issue tracking platform to enable the tailoring of data generated by CI-tools that operate on top of the VCS and issue tracking platform. We introduce so-called *SQA-Profiles* to describe the characteristic activity patterns of stakeholders within a certain role. For example, the project management committees (PMCs) of Apache projects are groups of contributors, who lead the project's development and community. The size of PMCs varies between 9 (Apache Jena) and 55 (Apache Httpd) members.[1] Despite all PMC members having the same formal roles, the actual task focus varies substantially between stakeholders. For example, one PMC member might take care of patch integration, while another handles issue management.

The aim of our approach is the establishment of a model for a project-independent definition of stakeholder profiles based on activity data. We analyze the activities of PMC members from 20 Apache projects, and derive a set of SQA-Profiles for PMCs. We extract the last year's project histories from the VCS and the issue tracking platform, and use a k-means clustering to categorize the activity data, and to derive rules for the definition of SQA-Profiles based on the characteristics of each resulting cluster. Additionally, we introduce a nominal scale of activity data with the values *High*, *Medium*, and *Low* to enable a project-independent and human-readable rule definition. For example, the cluster with a high merge activity and a medium or high commit activity forms the foundation for a profile describing the work of a stakeholder that integrates patches. The resulting set of SQA-Profiles covers *Bandleaders*, *Integrators*, *Gatekeepers*, and *Onlookers*.

The main contributions of this paper are as follows:

- **A model to describe activity profiles of stakeholders in a project-independent manner.**
- **A set of project-independent PMC member activity profiles**.
- **A framework to automatically profile stakeholders based on activity data mined from the VCS and the issue tracking platform.**

We implemented a prototypical framework (*SQA-Profiler*) to evaluate the performance of our rule-based approach. In this

[1] http://people.apache.org/committers-by-project.html

978-1-4799-8469-5/15 © 2015 IEEE

SANER 2015, Montréal, Canada

evaluation, we investigated whether a rule-based and project-independent approach, such as the one used by SQA-Profiles, can achieve a similar performance as a machine learning based approach, which has to be individually parameterized for each project. The results show that our approach can indeed determine profiles of PMC members with a precision of 0.92 and a recall of 0.78.

The remainder of this paper is structured as follows. In Section II we present the methodology we followed in this paper. In Section III, we introduce our rule-based approach and a set of SQA-Profiles, followed by the evaluation in Section IV. We discuss results in Section V, and threats to the validity of our research in Section VI. The most relevant related work is discussed in Section VII. Finally, we conclude with our main findings in Section VIII.

II. Approach

The aim of our approach is the profiling of stakeholders within a PMC. In Apache projects, the membership in a PMC is treated as role.[2] In comparison to other roles, such as contributor, the covered spectrum of tasks is broader in a PMC. Project committers work on issues and contribute source code changes. PMC members actively contribute issues and source code as well, but the PMC is additionally in charge of project and community management. The management of the project incorporates tasks, such as monitoring or gatekeeping. A benefit of using a committee compared to a single manager is the ability to share tasks among the different committee members. However, the resulting different focus of the PMC members requires a different view on the data presented in dashboards as well [1].

The extracted profiles can be used for an automatic composition of views or for a tailoring of CI-data in accordance to the activities of a PMC member. We address the goal of our approach with the following research questions:

RQ1: *Can activity data mined from the version control system and issue tracking platform be used for the extraction of profiles within a PMC?*

RQ2: *What profiles of PMC members can be extracted from the activity data, and how can these profiles be described in a ruled-based model?*

To answer these research questions, we studied the activity data of PMC members from 20 Apache projects between September 2013 and September 2014. All selected projects are Java projects that use Maven as build tool. We decided to analyze the time-range of one year instead of the entire project history to minimize the noise introduced by PMC member changes. The extracted activity data include the project name, the stakeholder associated with the activity, and the number of each of the following events: commits, merges, issue status changes, issue comments, issue assignee changes, and issue priority changes. These events are referred to as *attributes* in the remainder of the paper. In total, we ended up with 8'707 data points extracted from the VCS[3] and the issue tracking platform[4] of the according projects.

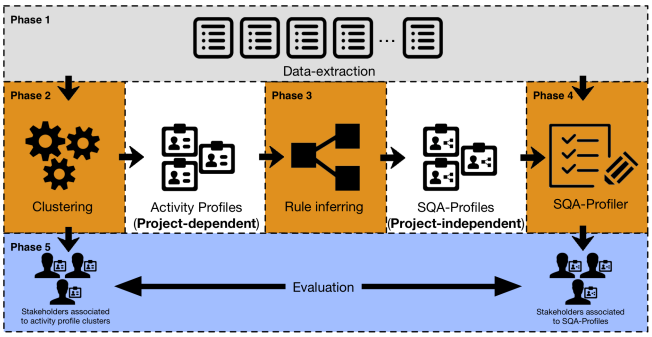

Fig. 1. Overview about the five phases of our approach

Figure 1 depicts the five phases of our approach:

Phase 1 - Data extraction
Extracting activity data from the VCS and the issue tracking platform.
Result: Stakeholder activity data records.

Phase 2 - Clustering
Clustering the extracted activity attributes.
Result: Stakeholders clustered based on their activity data. The clusters were computed based on the numerical activity values.
Remark: These values are **project-dependent** and can slightly vary between different projects.

Phase 3 - Rule inferring
Inferring project-independent activity profiles from the clusters generated in Phase 2. A nominal scale is used as abstraction layer to introduce project-independent values for the rule definition.
Result: A set of project-independent and human-readable activity profiles, called *SQA-Profiles*.
Remark: These values are **project-independent**.

Phase 4 - SQA-Profiler
Executing *SQA-Profiler*, based on the extracted activity data (Phase 1) and the derived SQA-Profiles (Phase 3).
Result: Associations of stakeholders to a SQA-Profile.

Phase 5 - Evaluation
Evaluating the results of the profile association process of Phase 2 and Phase 4. The two association processes are different as in Phase 2 the associations are computed based on project-dependent activity profiles, and in Phase 4 the associations are computed based on project-independent activity profiles.
Result: The performance of the proposed SQA-Profiles approach compared to a project-dependent baseline.

The aim of Phase 3 is to abstract activity profiles into rules to mitigate the discussion of thresholds needed for the definition of activity profiles. For example, it is hard to define a numerical and project-independent threshold for a "high" commit activity. Therefore, we use a nominal scale computed by machine learning to define high, medium, and

[2] http://www.apache.org/foundation/how-it-works.html#roles

[3] https://github.com/apache

[4] https://issues.apache.org/jira/

low values for each project individually. This adds a layer of abstraction, which enables a project-independent and human-readable definition of activity profiles.

In the evaluation, we compare the stakeholder profile associations computed by our rule-based approach against a baseline computed by machine learning. We decided to use a ground truth computed with machine learning instead of survey data, as the survey data might not be as objective as the values in the repository. We are aware that not all roles must be visible in the activity data of VCS and issue tracking platforms (see [4]). However, what we want to address with our approach is the circumstance that the self-described role or profile of a stakeholder can deviate from the actual role or profile.

III. Activity Profiling

Next, we describe each phase of our profiling approach.

A. Phase 1 & 2 - Data Extraction and Clustering

We merged the activity data extracted from the VCS and the issue tracking platform based on the user accounts, and combined those activities that are associated with the same email address. However, we noticed that some developers used different email addresses for their accounts in the VCS and the issue tracking system. To address this issue, we applied a matching algorithm that merges data from the issue tracking system with data from the VCS. The matching algorithm investigates the local part and the domain part of each email address separately, and merges the two data points if the local parts accord with each other. For example, we noticed that an account *username@gmail.com* is used in the issue tracking system, but not in the VCS, whereas the account *username@apache.org* is used in the VCS, but not in the issue tracking system. In this case, the matching algorithm merges the data extracted from the VCS with the data from the issue tracking system, since the local parts of the email addresses match. To avoid as many false merges as possible, the matching algorithms only merges two accounts, if one account is exclusively used for the VCS, while the other account is exclusively used for the issue tracking system.

In a second step, we acquired a list of all PMC members' repository accounts from the Apache website[5], and subsequently filtered out all the activity data that we could not associate with any PMC member. Furthermore, we also filtered out all PMC members that could not be associated with a total relative project activity of greater than zero percent, to remove PMC members that can not be classified due to missing data. The resulting absolute threshold for this activity filtering depends on the overall number of activities in the according software project. After these two preprocessing steps, our activity dataset contained 542 entries in total that could be associated to 130 different PMC members.

Table I provides an overview about the projects we used in the study and the number of active PMC members associated with each project, as well as the number of events related to assignee, comment, commit, merge, priority and status changes.

[5]http://people.apache.org/committers-by-project.html

TABLE I. Overview of the Apache projects used for our analysis with the active PMC members in each project, as well as the number of assignees, comments, commits, merges, issue priority, and status change events.

Project	# Assignee	# Comment	# Commit	# Merge	# Priority	# Status	# active PMCs
Accumulo	890	10681	1804	1292	154	2050	14
ActiveMQ	234	2784	1062	8	69	587	11
Ambari	273	9355	5008	28	55	7049	20
Camel	537	3291	3929	103	66	913	13
CXF	178	2612	3762	346	17	1443	13
Drill	909	3326	805	3	106	1716	11
Felix	231	2189	1000	0	7	1082	14
HBase	1651	69737	4214	3	441	11422	25
Hive	1126	32052	2162	95	126	9257	12
Jackr.-Oak	525	6063	3874	3	101	2408	18
Jena	99	1356	995	1	34	436	5
Karaf	462	2855	2028	46	33	871	7
Log4j2	140	3178	1918	18	33	579	7
Mahout	191	5072	366	5	22	1258	6
PDFBox	240	6074	1346	0	140	1193	8
Sling	398	3636	4417	2	20	1395	12
Spark	890	6240	4643	875	292	1949	17
Stanbol	50	569	542	0	7	221	6
Tika	48	2066	345	3	4	255	13
TomEE	39	604	1460	13	5	454	2
Total	*9111*	*173740*	*45680*	*2844*	*1732*	*46538*	*234*

To generate a set of profiles, we applied k-means clustering to our data, since it is efficient and computationally cheap in handling large datasets. To perform the clustering, we used Weka [5], a machine learning framework written in Java. For the clustering, we used the default settings of Weka for k-means, but we did not consider all the seven attributes we retrieved from the repositories. As the changes for the *fixed in version* attribute strongly correlated with the *status* ($\rho = 0.71$) and the *priority* attribute ($\rho = 0.72$), we removed this attribute from all further analyses. Using the k-means clustering algorithm, we constructed four clusters, as a closer analysis of different number of clusters has shown that this is the number of clusters that is neither too low so that we end up with heterogenous profiles, nor too high so that we end up with many different profiles with only minimal differences. However, it might be possible that analyzing other projects than the ones we selected for the study might lead to different clusters. Table II provides an overview about the characteristics of the four clusters we mined with Weka, as well as the characteristics of the entire data set. For each attribute and cluster combination, the table displays the centroid, indicating that there are big differences between some of the clusters. The table also shows the number of PMC members that are part of each cluster.

TABLE II. Centroids of the four activity profile clusters as well as the whole dataset, and the number of instances that were classified into each cluster.

Attribute	Full data	C1	C2	C3	C4
Commit	10.98	70.00	28.89	19.42	6.84
Merge	8.63	88.33	68.56	2.08	2.03
Status	9.25	64.67	9.56	34.92	4.75
Comment	6.97	42.67	3.67	22.08	4.53
Assignee	8.38	60.33	5.33	25.33	5.25
Priority	6.80	69.00	3.22	30.92	2.61
# Instances	*130 (100%)*	*3 (2%)*	*9 (7%)*	*12 (9%)*	*106 (82%)*

The *C4* cluster is by far the biggest cluster, followed by *C3*, *C2*, and *C1*. The differences in the cluster size are caused

by the characteristics of a certain profile within a software project. For example, the number of stakeholders that integrate source code changes into the main code base is restricted by the number of contributed patches. These four clusters provide the ground truth for the evaluation in Section IV and the basis for the definition of the rule-based profile model described in Section III-B.

B. Phase 3a - Rule Inferring

A main goal of our research is to provide a rule-based profile description model to enable a project-independent profile analysis. Hence, we introduce a model called *SQA-Profiles*, which is used for the description of the set of profiles. The SQA-Profile model uses a nominal scale to formulate rule-based profiles about the activities of a stakeholder within a software development tool (e.g., VCS, issue tracking platform). We decided to use a nominal scale instead of relative values to foster the readability of the rules for all stakeholders of a software project. From our perspective, the understanding of a rule is especially important in case of an automatic change in the profile association, because a stakeholder might want to know why she was associated to another profile.

We determined the nominal scale based on the mined activity data. In a first step, we normalized the absolute numbers of activities of each single attribute within the projects to relative values from 0 to 1. Secondly, we plotted the relative attribute values of all projects and based on an initial visual analysis, we clustered the relative values with k-means into three clusters. To ensure to get the best fitting classification, we additionally run the k-means clustering for two and four clusters. The results showed that the initial number of three clusters is the most appropriate one for a scale across all attributes. We decided to label the resulting clusters with commonly used names: *High*, *Medium*, and *Low*.

In addition to the nominal scale, a set of functions and logical operations is used for the rule definition. The model supports the basic logical operations *and* and *or*. We use following definitions and functions in the model:

- ***H(attribute), M(attribute), L(attribute), N(attribute)***: Functions to prove if a passed attribute has a nominal value of (H)igh, (M)edium, (L)ow, or (N)o value
- ***A***: The set of all attribute names (commit, merge, status, comment, assignee, priority)
- ***SH***: The set of all stakeholders

Table III provides an overview about the converted nominal values for each of the four clusters found in Phase 2 (see Table II). These values form the basis for our set of SQA-Profiles.

TABLE III. NOMINAL VALUES FOUND FOR EACH OF THE FOUR SQA-PROFILES.

Attribute	C1	C2	C3	C4
Commit	H	L/M	M	L/M
Merge	H	H	N/L/M/H	L/M
Status	H	L/M	H	L/M
Comment	H	L/M	N/L/M/H	L/M
Assignee	H	L/M	M	N/L/M
Priority	H	L/M	N/L/M/H	L/M

Based on the introduced nominal scale, logical operations, functions and the definitions, it is possible to formulate the following exemplary SQA-Profile:

Name: Example Rule

Rule: $\{s \in HH : H(s.commit) \wedge N(s.status)\}$

$HH = \{s \in SH : |\{a \in A : H(s.a)\}| > 1\}$

This example rule describes stakeholders with the following profile: at least two activity attributes with a *High* value, one of the *High* values must be the commit activity, and no activity on the status attribute. We additionally use the according quantity operators of the defined logical operations to foster the readability of the proposed rules.

C. Phase 3b - Initial set of SQA-Profiles

Based on the characteristics found in the converted clusters (see Table III) and the SQA-Profiles model we derived the following project-independent profile definitions.

The Bandleader profile describes a PMC member that has a high activity in each attribute. We call it *Bandleader* because a stakeholder with this profile keeps the music playing in a project, and it is very likely that the music stops when such a stakeholder leaves the project.

We found three PMC members in three different projects with this profile. The projects are Apache Drill, Jena, and Karaf. In the Apache Drill project, the stakeholder with the Bandleader profile has ten times more commits than the stakeholder with the second most commits. The activity data of the other two projects shows a similar picture.

The SQA-Profile of the Bandleader is as follows:

Name: Bandleader

Rule: $\{s \in SH : |\{a \in A : H(s.a)\}| = (|A|)\}$

The Integrator profile describes a PMC member that has a high merging activity in the VCS, and at least one other attribute with moderate activity. We call this profile *Integrator*, because a stakeholder with this profile primarily handles the integration of source code contributions in a software project. As part of this activity, source code has to be integrated in the VCS, and a change has to be noted in the according issue (e.g., status change or comment).

We found nine PMC members in nine different projects with the Integrator profile. The projects are Apache Accumulo, ActiveMQ, Camel, CXF, HBase, Hive, Jackrabbit-Oak, Sling, and Spark. None of these projects has a stakeholder associated with the Bandleader profile.

The SQA-Profile of the Integrator is as follows:

Name: Integrator

Rule: $\{s \in HH \cap HM : H(s.merge)\}$

$HH = \{s \in SH : |\{a \in A : H(s.a)\}| > 0\}$

$HM = \{s \in SH : |\{a \in A : M(s.a)\}| > 0\}$

The Gatekeeper profile describes a PMC member that has high activity in status changes and a moderate activity in assignee changes or commits. We refer to this profile as *Gatekeeper* for a stakeholder who decides when the status of an issue gets changed. We were able to find two variations of this profile. The difference between the variations is the activity in the attributes assignee and commit. In some projects, stakeholders of this profile mainly take care of the gatekeeping on issue level. In other projects, stakeholders of this profile have a broader focus, and make changes in source code contributions or actively contribute own source code changes as well.

We found twelve PMC members in nine different projects with the Gatekeeper profile. The projects are Apache ActiveMQ, Camel, Felix, HBase, Jackrabbit-Oak, Mahout, PDFBox, Sling, and Stanbol. None of these projects has a stakeholder associated with the Bandleader profile, but five projects have a stakeholder associated with the Integrator profile as well. This can indicate that projects with a stakeholder associated to the Gatekeeper profile also have a stakeholder associated to the Integrator profile.

The SQA-Profile of the Gatekeeper is as follows:

Name: Gatekeeper

Rule: $\{s \in AA \cup AC : H(s.status)\}$

$AA = \{s \in SH : H(s.assignee) \vee M(s.assignee)\}$

$AC = \{s \in SH : H(s.commit) \vee M(s.commit)\}$

This rule covers both variations (assignee changes, commits). The threshold for one of both variations is defined with a *High* or *Medium* activity in the according attribute.

The Onlooker profile describes a PMC member that only occasionally contributes to the VCS and the issue tracking platform of a project. The sporadic activity in VCS and issue tracking platforms make it hard to define a rule for this profile. We use the term *Onlooker* because, from the perspective of the VCS and issue tracking platform, their contribution is limited. However, it can be that the according stakeholders are more focused on the non-technical part of project management, such as community management. We found 106 PMC members almost equally distributed across all projects with the Onlooker profile.

The SQA-Profile of the Onlooker is as follows:

Name: Onlooker

Rule: $\{s \in M1 \cup (M0 \cap L1) \cup (L1 \cap NA)\}$

$L1 = \{s \in SH : |\{a \in A : L(s.a)\}| > 1\}$

$M0 = \{s \in SH : |\{a \in A : M(s.a)\}| > 0\}$

$M1 = \{s \in SH : |\{a \in A : M(s.a)\}| > 1\}$

$NA = \{s \in SH : N(s.assignee)\}$

We were not able to extract a clear activity pattern for this profile, but we found out that stakeholder with this profile have a certain level of activity in multiple attributes. Therefore, we described this profile with three variations addressing the activity level. The first variation addresses stakeholders that have at least two attributes with a *Medium* activity. The second variation addresses stakeholders that have at least one attribute with a *Medium* activity and at least two attributes with a *Low* activity. The last variation addresses stakeholders that have at least two attributes with a *Low* activity and no activity on the assignee attribute.

D. Phase 4 - SQA-Profiler

The proposed nominal scale and the SQA-Profiles enable an automatic processing of software development activity data. We implemented a framework called *SQA-Profiler* to automatically extract stakeholders with an activity history that matches one of the defined SQA-Profiles.

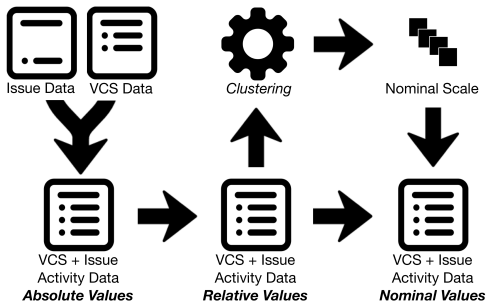

Fig. 2. Dataflow in the SQA-Profiler

Figure 2 depicts the dataflow in the SQA-Profiler framework. The framework expects stakeholder records with absolute activity data as input (e.g., one commit, five comments, no merge). It also supports an automatic merging of incomplete data sets (e.g., in case that a stakeholder uses different email addresses in the VCS and the issue tracking platform). In a second preprocessing step, the absolute activity data gets normalized per project to compute the borders of the nominal values. Afterwards, every relative value gets transferred into the according nominal value. The resulting nominal values are used for the evaluation against the proposed set of SQA-Profiles.

SQA-Profiler evaluates each data set against the rules of the profiles specified in Section III-C. The evaluation goes from the most specific profile (the Bandleader) to more generic ones (the Onlooker). The first matching rule stops the evaluation process, and classifies the data set with the according profile. A data set is marked as unclassified in case that no rule of no profile matches it. The rules and the according evaluation are hard-coded at the current stage for simplicity reasons. In future versions, SQA-Profiler will offer a domain specific language for rule specification.

The output generated by SQA-Profiler is a list of stakeholders with their SQA-Profile based on the activity data. The SQA-Profiler is available for download on our project website.[6]

IV. Evaluation

A central claim of our approach is that rule-based activity profiles, such as SQA-Profiles, can be used to establish project-independent profile definitions. State-of-the-art approaches,

[6]http://www.ifi.uzh.ch/seal/people/brandtner/projects/sqa-profiles.html

such as machine learning, provide a powerful tooling to cluster data precisely, but it is hard to define project-independent profile definitions. In this evaluation, we compare the results of the automatic profile association provided by SQA-Profiler with the semi-automatic profile association based on clusters provided by Weka. Figure 3 depicts a simplified overview of our evaluation method.

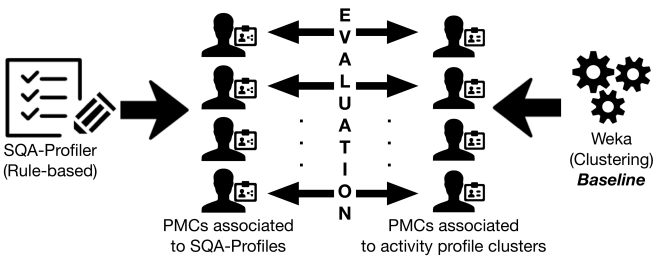

Fig. 3. Evaluation - Overview

The input data for both approaches is a list of PMC members and their associated activity data. In case of the SQA-Profiler approach, the initial input data contains data sets of non-PMC members as well. The filtering of PMC members takes place after the preprocessing steps. This is necessary, because the transformation of the absolute activity attribute values to nominal values takes place in the preprocessing steps. A transformation without the activity data from non-PMC members would distort the nominal values. Figure 4 depicts the data-flow of the SQA-Profiler in this evaluation setting.

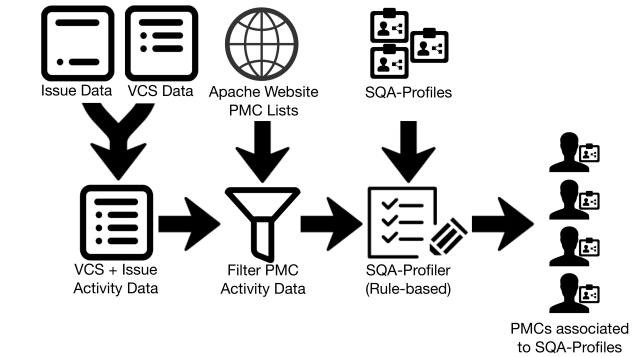

Fig. 4. Evaluation - Data flow

We ran the evaluation on the activity data of 20 Apache projects and automated the evaluation process to cope with the large amount of data. An evaluation program starts (1) a Weka instance for clustering, (2) a SQA-Profiler instance to associate profiles, and (3) compares the stakeholder-profile associations per project. The Weka instance is started with a data set that was manually preprocessed up-front. The preprocessing incorporates the merging of different identities used in the VCS and issue tracking platform. The SQA-Profiler instance uses raw activity data extracted from the VCS and issue tracking platform as input.

Table IV lists the precision, recall and F-measure achieved by our automatic and rule-based approach compared to the semi-automatic baseline with machine learning. A *true-positive* (TP) is any stakeholder-profile association that is in accordance with the classification of the baseline dataset and a *false-positive* (FP) is any stakeholder-profile association that is not part of the baseline dataset.

TABLE IV. RULE-BASED CLASSIFICATION - PERFORMANCE

Profile	TP	FP	Total	Precision	Recall	F-measure
Bandleader	3	1	3	0.75	1.00	0.86
Integrator	9	1	9	0.90	1.00	0.95
Gatekeeper	9	5	12	0.64	0.75	0.69
Onlooker	80	2	106	0.98	0.75	0.85
Overall	*101*	*9*	*130*	*0.92*	*0.78*	*0.84*

In total, our approach classified 101 stakeholders correctly (*true-positive*), 9 stakeholders to a wrong profile (*false-positive*), and 20 stakeholders kept unclassified (*false-negatives*). These results lead to an overall precision of 0.92 and a recall of 0.78 compared to the baseline.

The Integrator profile achieved the best result with a precision of 0.90 and a recall of 1. Followed by the Bandleader and the Onlooker profile with a precision of 0.75 and a recall of 1, and a precision of 0.98 and a recall of 0.75, respectively. The Gatekeeper profile has a precision of 0.64 and a recall of 0.75, which leads to a relatively low F-measure (0.69) compared to the other profiles. A reason for this low precision can be the broad definition of this profile caused by different gatekeeping processes of different software projects. For example, in some projects the Gatekeeper changes the status and the assignee, whereas in other projects the Gatekeeper has to additionally apply the patches. The same reason affects the Onlooker profile. Another interesting point are the *false-positive* matches in the Bandleader and the Integrator profile. These two false-positives a very likely caused by the blurring, which was introduced with the conversion from numerical to nominal values in the SQA-Profiles. Based on the nominal attribute values the profile association is correct, but based on the numerical values the matches are wrong.

Additionally, we evaluated the distribution of nominal values to ensure that they are equally distributed. This is important since the proposed nominal scale determination approach does not explicitly address this issue. The chart on the left in Figure 5 depicts the distribution of the nominal values extracted from the activity rating of all stakeholders. Despite the share of activities with ”no rating” is larger than all other shares, the figure does not indicate any unequal distribution introduced by the nominal scale.

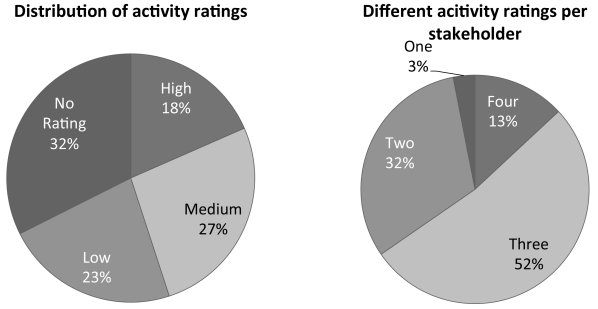

Fig. 5. Distribution of activity ratings and number of different activity ratings per stakeholder

The second chart in Figure 5 depicts the number of different activity ratings per stakeholder. This chart shows that three percent of the analyzed PMC members have rated all attributes with the same value. An example profile for a stakeholder with *High* rating in all attributes is the Bandleader. The number of different activity ratings per stakeholder is an important value for the interpretation of the evaluation results. For example, three of the proposed SQA-Profiles have a restriction on the minimum number of a certain attribute value (e.g., more than two attributes with a *Low* value). The result show that 65% of the stakeholders have more than two different activity ratings. This can be an indicator that the number of *false-positives* can be reduced by the use of more attribute restrictions in the according SQA-Profile definition.

We additionally evaluated the distribution of the established SQA-Profiles between PMC members and stakeholders that are not part of a PMC. Table V shows the projects, in which a certain profile was found and whether or not the stakeholder is a PMC member.

TABLE V. SQA-PROFILES PER APACHE PROJECT

Profile	PMC Member	No PMC Member
Bandleader	Drill, Jena, Karaf, Spark	-
Integrator	Accumulo (2), ActiveMQ, Camel, CXF, HBase, Hive, Jackrabbit-Oak, Sling, Spark	Ambari, Log4j2, Mahout, TomEE
Gatekeeper	ActiveMQ, Camel, Felix (3), HBase, Hive, Jackrabbit-Oak, Mahout, PDFBox, Sling, Stanbol, Tika (2)	Accumulo (2), Ambari (2), Camel, CXF, Felix (2), Log4j2 (2), Spark, Tika, TomEE (2)
Onlooker	all projects	all projects

The results showed that stakeholders with the Bandleader profile are always members of the PMC. In case of the Integrator profile, the majority of the found stakeholders are PMC members as well. However, in four projects the Integrator profile is associated with a stakeholder that is not a PMC member. In all of these four projects, no PMC member was associated with the Integrator profile. With exception of the Apache Accumulo project, each project has exactly one stakeholder with this profile. The Gatekeeper profile is associated with stakeholders with and without PMC membership. Different to the Bandleader and the Integrator profile, this profile is assigned to multiple stakeholders independent of their PMC membership. For example, in the Apache Felix project the Gatekeeper profile is associated with three PMC members and with two stakeholders without PMC membership. The Onlooker profile was found in all of the analyzed projects independent of a stakeholder's PMC membership. The found stakeholders are almost equally distributed across all projects.

V. DISCUSSION

Overall, we found evidence that activity data mined from the VCS and the issue tracking platform can reflect the tasks of stakeholders within a certain role. The evaluation results showed that the rule-based SQA-Profile approach performs almost as good as the baseline approach using machine learning. These results indicate that our automatic and rule-based approach can achieve similar results as a semi-automatic and project-dependent approach. We discuss the benefits of a rule-based approach and a number of factors that can influence the performance of SQA-Profiles.

A. Nominal Scale & Rule-based Profiles

The proposed nominal scale and rule-based profiles provide a simple yet powerful model to describe stakeholder profiles. We showed that, despite this simplicity and the project-independent definition, the SQA-Profiles approach performs almost as good as machine learning using precise values.

From our perspective, it is important to keep the rules simple and comprehensible for stakeholders such as project managers. The rules have to be transparent and easily adoptable, because even a perfect profiling approach can produce results that are not rational from the perspective of a certain software project. This is important, because missed information in software development influences the decision quality and the project budget. Especially, a changing focus and changing activities of a stakeholder during her work on a software project can lead to non-optimal results. For example, the amount of patches increases shortly before a feature freeze deadline. In such a case, an additional PMC member might help out with the patch integration. The patch integration activity can influence the profile association (e.g., the associated profile changes from Gatekeeper to Integrator). In case of such a profile change, it is important that a stakeholder can follow the profile association process and adopt the profile, if necessary.

B. A Set of SQA-Profiles

The set of SQA-Profiles proposed in this work covers activity patterns of PMC members in 20 Apache projects over the last year. There are indicators that the set of rule-based profiles does not cover all profiles in any PMC of a software project.

The extracted activity dataset of some PMC members is relatively small or empty. For example, there are PMC members with only one comment or only one commit within the analyzed time range. Furthermore, for a small groups of PMC members, we were not able to find any activity in the VCS or the issue tracking platform. We can imagine two scenarios for the absence of activities. The first scenario is that the according PMC members no longer participate in the development of the project. The second scenario is that the according PMC members are in charge of community related tasks (e.g., management of mailing lists, wikis) and, therefore, do not contribute to the VCS or issue tracking platform. We decided to not cover such scenarios in our set of SQA-Profiles, as it is very likely that such PMC members are not interested in CI-data.

C. Project Organization

The evaluation results showed that the existence of profiles and the number of stakeholders associated with a profile within a software project is influenced by the project's organization. For example, in projects with a stakeholder associated to the Bandleader profile, no further stakeholder was found with an Integrator or Gatekeeper profile. This is interesting because the SQA-Profile model evaluates each stakeholder independently from each other. Theoretically, it would be possible that a project has a stakeholder associated to the Bandleader profile and another stakeholder associated to any other profile. Based on the analyzed projects, a stakeholder with the profile

Bandleader is an indicator that a project has a relatively small *truck factor* [6]. The *truck factor* indicates the number of stakeholders that have to be hit by a truck before the projects gets incapacitated.

The results further showed that the existence of a stakeholder with the Integrator profile indicates when a software project has a dedicated source code integration process. However, the absence of a stakeholder with this profile does not indicate the absence of a dedicated integration process. In some cases, contributors hand-in patches as attachment in the issue tracker. The integration of such patches is typically done by a PMC member and from the perspective of the VCS it is hard to differentiate them from normal source code commits.

D. Project Relationships

The analysis of activity data from the VCS and issue tracking platform showed that some of the Apache projects have a strong relationship. We found relationships on technical level and on PMC member level.

We found one stakeholder that is PMC member and contributor in various Apache projects. He also appears in the PMC of five projects analyzed in this work: ActiveMQ, Camel, CXF, Felix, and Karaf. The link between the mentioned projects is the Apache ServiceMix project, which combines the mentioned projects to an integration container and where the stakeholder is PMC member as well. Despite his PMC memberships in these five projects, the stakeholder contributed only to the Apache Felix and Karaf project in the last year.

Entries in the issue tracking platform indicate that there are technical dependencies of Apache projects as well. For example, issue entry *FELIX-4436* in the Felix project is caused by an improvement in the ServiceMix project described in issue entry *SMX4-956*. Furthermore, the issue entry *KARAF-2420* in the Karaf project is related to the mentioned improvement. The initial issue entry in the Apache ServiceMix project requests an improvement in the monitoring of changes in a configuration file. This example shows that relatively simple changes in one project can affect multiple other projects.

In our work, we evaluated the performance of SQA-Profiles on project-level. We profiled activity data of stakeholders per project and independent of any activity in other projects. In order to support project-overlapping profiles and change tracking, it would be necessary to adopt the profile association algorithm and to derive additional SQA-Profiles.

E. Contributors with PMC Profiles

In our evaluation, we extracted a number of non-PMC members in Apache projects, which have an activity history matching PMC profile (see Table V).

We found three projects that have associated the Integrator and Gatekeeper profile only to stakeholders that are not members of the PMC: Apache Ambari, Log4j2, and TomEE. In case of the TomEE project, the activity of the PMC members in the VCS and the issue tracking platform is relatively low. We could only find activities of two PMC members. Most of the source code contributions originate from the contributors. The contributors also take care of the patch integration and the management of the issue tracing platform. The analysis of the Log4j2 project draws a similar picture. In case of the Ambari project, the situation is different. The PMC of the Apache Ambari project consists of 37 stakeholders, which is large compared to other Apache projects. Only ten out of 47 contributors are not PMC members. However, the contributions of the PMC members seem to be limited because the Integrator and Gatekeeper profile are associated to stakeholders that are only listed as contributor.

The existence of non-PMC members with a PMC profile can be seen as indicator that the roles assigned in a software project do not always reflect the actual activity of a stakeholder. This finding impacts our proposed approach in the definition process of SQA-Profiles, because wrongly assigned stakeholders may blur the resulting SQA-Profiles.

F. View Composition and Information Tailoring

The motivation of this work originated from the idea to automatically compose views and tailor information for CI dashboards based on activity profiles of stakeholders.

We showed that it is possible to extract activity patterns from the data of a VCS and an issue tracking platform. These patterns can be used to establish rule-based profiles for an automatic processing. The evaluation of 20 Apache projects further showed that a stakeholder profile described with our proposed rule-based model is project-independent.

However, the data-driven objective of our approach has limitations as well. A major limitation is the classification of profiles that describe stakeholders with a low activity in the used data sources (e.g., VCS, issue tracking platform). This limitation is reflected in the relatively large Onlooker profile. Due to the small activity it is neither possible to further split up the cluster nor to extract a significant activity pattern. One possibility to overcome this limitation would be to raise the threshold of the minimum activity of a stakeholder that is required to enable the profiling. In general, we propose a generic view for all stakeholders that have no profile associated, because of a low activity. Another limitation is the assumption that activity data reflects the importance of information. For example, a change in a source file causes an error in another file, which was never touched by the according stakeholder. From the perspective of the activity data, the information about the changed source file is more important to the stakeholder than information about the other file. One approach to overcome this would be adding structural information, such as source dependencies, to the model.

Overall, we see our approach as a milestone to enable a fully-automatic data processing for information tailoring and view composition in CI environments.

VI. Threats to Validity

Empirical studies have limitations that have to be considered when interpreting their results. Our study is amenable to threats to the external, internal and construct validity.

External Validity. For the extraction of the rule-based SQA-profiles, we relied on activity data gathered by mining source code repositories and issue tracking platforms of 20 different Apache projects. We limited the activity data extraction to a

period of one year. These decisions might limit the generalizability of our results, and further studies might need to be conducted to verify that our results can also be applied to other projects. However, to mitigate this risk, we have chosen the projects used in our analysis in a way to get a broad sample of various projects with different characteristics.

Internal Validity. For the evaluation of the rule-based activity profiles, we first used Weka and applied clustering to the activity data to get four clusters that we then used as the ground truth. Thus, the clusters might only be an approximation of the real activity profile of each stakeholder, which can affect the results of our evaluation. We mitigated this risk by verifying that the clusters are sound, that is, the data in the clusters are similar to each other while dissimilar to data of other clusters.

As another threat to the internal validity, we did not differentiate between the various projects that we used in our study and applied the same approach to all the projects. However, as the results have indicated, there are for example projects that adhere to the Apache guidelines that state how a program committee should work, while others do not. By taking these differences into account, we might be able to improve our results even more.

Construct Validity. The major goal of our approach is to establish a rule-based approach to automatically extract SQA-Profiles. This rule-based approach relies on commit and issue management activities of stakeholders involved in a software project. A threat to the validity of the study is that there might be other factors than the commit and issue management activities that have an influence on a stakeholder's focus within a certain role, which are currently not captured by our approach. Further studies need to be conducted to examine the influence of these yet unknown factors. Another potential threat of our proposed approach is the partial re-use of the activity data for the profile extraction and for the evaluation. We tried to mitigate this threat by using only the activity data of PMC members for the profile extraction and the activity data of all members of an Apache project for the evaluation.

VII. Related Work

The proposed approach of extracting stakeholder profiles based on activity data can be seen as intersection of multiple research areas. In the following, we discuss the most relevant related work from the following areas: socio-technical networks, bug prediction, and developer context.

Socio-technical networks: Bird et al. [7] mined communication and development data, and found that strong community structures exist in the communication patterns of open source projects. Surian et al. [8] investigated patterns in a graph-based representation of developer interactions. They used the found patterns to establish a recommendation for finding developers with similar properties [9]. Another approach proposed by Meneely et al. [10] aims to enrich the data gathered from VCS with issue tracking annotations. Their results showed that some groups of contributors never appear in the VCS, but actively influence the development process. A further topic of this research area is social coding. Dabbish et al. [11] investigated the influence of visible feedback on the collaboration of community members. They indirectly categorized the roles of developers based on attributes, such as number of followers or commenting activity. The related research in socio-technical networks showed that different attributes (e.g., number of commits or comments) from multiple repositories (VCS, issue tracker, etc.) can be used to successfully model the interactions of developers within a software project.

Bug prediction: Antoniol et al. [12] proposed an approach to classify and distinguish bugs based on the information in the issue description. They used an alternating decision tree to predict the type of an issue. Guo et al. [13] investigated a characterization of bugs to predict which of them get fixed. For example, they showed that the number of reassignments negatively influences the likelihood for a bug fix. Zimmermann et al. [14] categorized the bug reopen process based on quantitative bug data (e.g., state, assignee, type) and survey data where they asked about reasons for a reopening. Ostrand et al. [15] introduced a negative binomial regression model to predict the expected number of failures within a source code file. They used the fault and modification of previous releases for their predictions. Weyuker et al. [16] extended the approach of Ostrand et al. with developer information. They derived metrics addressing the number of developers, which modified a file. Pinzger et al. [17] investigated the fragmentation of developer contributions and the number of post-release failures. They established a contribution network and showed that centrality measures, such as number of authors and commits can predict failure-prone binaries with a high precision. The relation of this research area to our approach is the systematic analysis of attributes from repositories to derive rules for bug prediction or in our case profiling.

Developer context: State-of-the-art integrated development environments (IDEs), such as Eclipse, provide various interface configurations for different roles (e.g., Java Developer, Web Developer). Findlater et al. [18] showed that a fine-grained and more task-oriented grouping of interface elements is more efficient compared to a single user interface composition per role. Cheng et al. [19] investigated the collaboration data of the Rational Team Concert platform. The collaboration data can be used to support the composition of personal user interfaces in IDEs. Another aspect besides the role is the task context for recommendation systems in IDEs. Kersten and Murphy [20] proposed Mylar (initial name of Mylyn) to track task contexts in the IDE. The proposed interest model of Mylar can help developers to stay focused on a task by highlighting important artifacts. Anvik and Murphy [21], [22] investigated the implementation expertise of developers based on the data of the VCS and the issue tracker. They came up with an automatic recommender system to support bug-triaging. Fritz et al. [23] introduced a degree-of-knowledge model to estimate the knowledge of a stakeholder about a certain source code artifact. They found that the code a developer authors and the code with which the developer interacts are not the same. Ying and Robillard [24] proposed techniques to store and process developer profiles for recommendation purposes. They reviewed existing recommendation approaches from movie databases and investigated potential applications in software engineering. The relation of this research area is the aim to describe the context of stakeholders within a software project.

Our approach differs from the mentioned related work, as we put the focus on the individual activities of stakeholders and not on the interactions between stakeholders.

VIII. Conclusion

CI environments have become an important information source in software development. In this paper, we introduced rule-based and project-independent *SQA-Profiles* as an instrument to support information propagation in software projects.

We analyzed the activity data of project management committee (PMC) members from 20 Apache projects and derived four SQA-Profiles: Bandleader, Integrator, Gatekeeper, and Onlooker. We implemented *SQA-Profiler* as a prototypical framework to support the automatic identification of stakeholders and SQA-Profiles based on VCS and issue tracking data. The analysis showed that reoccurring activity patterns associated with a certain task (e.g., patch integration) can be found across different software projects. However, the occurrence of these patterns is not always in accordance with the assigned role of the stakeholders.

In the evaluation, we compared the performance of our automatic approach against a semi-automatic analysis with machine learning. The results showed that our rule-based and project-independent SQA-Profiles can be used to automatically extract the profiles of PMC members with a precision of 0.92 and a recall of 0.78 compared to the dataset extracted by a project-dependent and semi-automatic approach based on machine learning.

The SQA-Profiles approach can be seen as a potential data source for future algorithms that enable automatic view composition and information tailoring in CI environments. In future work, we want to bridge the gap between the proposed SQA-Profiles and our SQA-Mashup approach for an automatic composition of CI dashboards based on stakeholder activity history. To achieve this aim, we will have to translate the focus of stakeholders described by SQA-Profiles to the according data (e.g., quality metrics, build status) provided by CI environments.

References

[1] R. P. L. Buse and T. Zimmermann, "Information needs for software development analytics," in *Proceedings of the 34th International Conference on Software Engineering*, 2012, pp. 987–996.

[2] M. Brandtner, E. Giger, and H. Gall, "Supporting continuous integration by mashing-up software quality information," in *IEEE Conference on Software Maintenance, Reengineering, and Reverse Engineering*, 2014, pp. 184–193.

[3] M. Brandtner, E. Giger, and H. Gall, "SQA-Mashup: A Mashup Framework for Continuous Integration," *Information and Software Technology*, 2014.

[4] J. Aranda and G. Venolia, "The secret life of bugs: Going past the errors and omissions in software repositories," in *Proceedings of the 31st International Conference on Software Engineering*, ser. ICSE '09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 298–308.

[5] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The WEKA data mining software," *ACM SIGKDD Explorations Newsletter*, vol. 11, no. 1, p. 10, 2009.

[6] L. Williams and R. R. Kessler, *Pair Programming Illuminated*. Addison-Wesley Professional, 2003.

[7] C. Bird, D. Pattison, R. D'Souza, V. Filkov, and P. Devanbu, "Latent social structure in open source projects," in *Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of software engineering*, 2008, pp. 24–35.

[8] D. Surian, D. Lo, and E.-P. Lim, "Mining Collaboration Patterns from a Large Developer Network," in *2010 17th Working Conference on Reverse Engineering*, 2010, pp. 269–273.

[9] D. Surian, N. Liu, D. Lo, H. Tong, E.-P. Lim, and C. Faloutsos, "Recommending People in Developers' Collaboration Network," in *2011 18th Working Conference on Reverse Engineering*, 2011, pp. 379–388.

[10] A. Meneely, M. Corcoran, and L. Williams, "Improving developer activity metrics with issue tracking annotations," in *Proceedings of the 2010 ICSE Workshop on Emerging Trends in Software Metrics*, 2010, pp. 75–80.

[11] L. Dabbish, C. Stuart, J. Tsay, and J. Herbsleb, "Social coding in GitHub," in *Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work*, 2012, p. 1277.

[12] G. Antoniol, K. Ayari, M. Di Penta, F. Khomh, and Y.-G. Guéhéneuc, "Is it a bug or an enhancement?" in *Proceedings of the 2008 conference of the center for advanced studies on collaborative research meeting of minds*, 2008, p. 304.

[13] P. J. Guo, T. Zimmermann, N. Nagappan, and B. Murphy, "Characterizing and predicting which bugs get fixed," in *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering*, vol. 1, May 2010, pp. 495–504.

[14] T. Zimmermann, N. Nagappan, P. J. Guo, and B. Murphy, "Characterizing and predicting which bugs get reopened," in *Proceedings of the 34th International Conference on Software Engineering*, 2012, pp. 1074–1083.

[15] T. Ostrand, E. Weyuker, and R. Bell, "Predicting the location and number of faults in large software systems," *IEEE Transactions on Software Engineering*, vol. 31, no. 4, pp. 340–355, 2005.

[16] E. J. Weyuker, T. J. Ostrand, and R. M. Bell, "Using Developer Information as a Factor for Fault Prediction," in *Third International Workshop on Predictor Models in Software Engineering*, 2007, p. 8.

[17] M. Pinzger, N. Nagappan, and B. Murphy, "Can developer-module networks predict failures?" in *Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of software engineering*, Nov. 2008, pp. 2–12.

[18] D. M. Leah Findlater, Joanna Mcgrenere, "Evaluation of a Role-Based Approach for Customizing a Complex Development Environment," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2008, pp. 1267–1270.

[19] L.-T. Cheng, C. R. de Souza, S. Hupfer, J. Patterson, and S. Ross, "Building Collaboration into IDEs," *Queue*, vol. 1, no. 9, p. 40, 2003.

[20] M. Kersten and G. C. Murphy, "Mylar," in *Proceedings of the 4th international conference on Aspect-oriented software development*, Mar. 2005, pp. 159–168.

[21] J. Anvik and G. C. Murphy, "Determining Implementation Expertise from Bug Reports," in *Fourth International Workshop on Mining Software Repositories*, 2007, p. 2.

[22] J. Anvik and G.C. Murphy, "Reducing the effort of bug report triage," *ACM Transactions on Software Engineering and Methodology*, vol. 20, no. 3, pp. 1–35, 2011.

[23] T. Fritz, J. Ou, G. C. Murphy, and E. Murphy-Hill, "A degree-of-knowledge model to capture source code familiarity," in *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering*, 2010, pp. 385–394.

[24] M. P. Robillard, W. Maalej, R. J. Walker, and T. Zimmermann, Eds., *Recommendation Systems in Software Engineering*. Springer Berlin Heidelberg, 2014.